IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Applicant:	Eric E. Lowe	
App. No.:	10/769,586	Con. No.: 8953
Filed:	January 30, 2004	Art Unit: 2189
Title:	METHOD AND APPARATUS FOR MEMORY MANAGEMENT IN A MULTI-PROCESSOR COMPUTER SYSTEM	Examiner: Gu, Shawn X.

REQUEST FOR CORRECTED FILING RECEIPT

Commissioner for Patents
Attn: Office of Initial Patent Examination's
Filing Receipt Corrections
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

It has been noted that the Filing Receipt dated June 16, 2004, contains an omission of the domestic priority information for this application. Please note that this application claims the benefit of U.S. Provisional Patent Application No. 60/537,431, filed on January 17, 2004.

The present application was filed thirteen days after the provisional application was filed. As such, a Filing Receipt for the provisional patent application had not been received at the time of filing the nonprovisional patent application. Therefore, the application number for the provisional patent application was not known and was omitted from the specification and thus, also the Filing Receipt of the nonprovisional.

A copy of the Filing Receipt with the correction noted thereon is attached. Also attached is a copy of the first page of the specification claiming priority to the provisional application with an omitted application number. Lastly, attached is a copy of the Amendment and Response to Office Action filed on June 19, 2006 inserting the provisional application number into the specification.

Applicant respectfully requests the issuance of a corrected filing receipt showing the domestic priority information of the application.

The Applicant believes no fees or petitions are due with this filing. However, should any such fees or petitions be required, please consider this a request therefor and authorization to charge Deposit Account No. 04-1415 as necessary.

Dated: 5 DEC 2006

Respectfully submitted,

[illegible]

Gregory P. Durbin, Registration No. 42,503
Attorney for Applicant
USPTO Customer No. 66083

DORSEY & WHITNEY LLP
Republic Plaza Building, Suite 4700
370 Seventeenth Street
Denver, Colorado 80202-5647
Phone: (303) 629-3400
Fax: (303) 629-3450

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPL NO.	FILING OR 371 (c) DATE	ART UNIT	FIL FEE REC'D	ATTY.DOCKET NO	DRAWINGS	TOT CLMS	IND CLMS
10/769,586	01/30/2004	2186	1190	SUN1P766/P040534	12	29	6

CONFIRMATION NO. 8953

022434
BEYER WEAVER & THOMAS LLP
P.O. BOX 778
BERKELEY, CA 94704-0778

FILING RECEIPT

OC000000012959932

Date Mailed: 06/16/2004

Receipt is acknowledged of this regular Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Filing Receipt Corrections, facsimile number 703-746-9195. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).**

Applicant(s)

Eric E. Lowe, Pflugerville, TX;

Assignment For Published Patent Application

Sun Microsystems, Inc.;

Domestic Priority data as claimed by applicant

This app. claims the ben. of 60/537,431, filed 01-17-2004

Foreign Applications

If Required, Foreign Filing License Granted: 06/15/2004

Docketed Date JUN 18 2004 KFM BERKELEY

CW 6/1

SDB/FAK

Projected Publication Date: Request for Non-Publication Acknowledged

Non-Publication Request: Yes

Early Publication Request: No

Title

Method and apparatus for memory management in a multi-processor computer system

Preliminary Class

711

LICENSE FOR FOREIGN FILING UNDER
Title 35, United States Code, Section 184
Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Office of Export Administration, Department of Commerce (15 CFR 370.10 (j)); the Office of Foreign Assets Control, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

METHOD AND APPARATUS FOR MEMORY MANAGEMENT IN A MULTI-PROCESSOR COMPUTER SYSTEM

Inventor:
Eric E. Lowe

BACKGROUND OF THE INVENTION

1. Related Application

The invention described herein relates to and claims priority from the Provisional Patent Application having U.S. Serial Number ___________ (Attorney Docket No. SUN1P766P/P040534), entitled "Method and Apparatus for Memory Management in a Multi-Processor Computer System", invented by Eric E. Lowe, filed on January 17, 2004. The aforementioned patent document is hereby incorporated by reference in its entirety for all purposes.

2. Background

The present invention relates to multiprocessor computing systems and, more particularly, to memory management unit (MMU) trap synchronization in multiprocessor computing systems. Multiprocessor computing systems are coming into increasingly common usage due to the many advantages inherent in such systems. In such systems, a single operating system controls the operation of all the microprocessors (CPU's) of the system. Common multiprocessor systems include scalable shared memory (SSM) system symmetric multiprocessor (SMP) systems. System symmetric multiprocessing can make use of multiprocessor computing architectures configured so that all CPU's can access all random access memory locations. Many architecture can implement such systems. Example include without limitation X86 systems as well as SSM system symmetric multiprocessor system architectures designed by Motorola, IBM, and Microsoft (e.g., NT systems). Linux based systems can also take advantage of the principles of the invention. Another architecture suitable for implementing the principles of the invention is the so-called SPARC architecture. SPARC is short for Scalable **P**rocessor **Ar**chitecture, a RISC (reduced instruction set computer) technology developed by Sun Microsystems. The

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Eric E. Lowe	Attorney Docket No.: SUN1P766/P040534
Application No.: 10/769,586	Examiner: GU, SHAWN X.
Filed: January 30, 2004	Group: 2189
Title: METHOD AND APPARATUS FOR MEMORY MANAGEMENT IN A MULTI-PROCESSOR COMPUTER SYSTEM	Confirmation No.: 8953

AMENDMENT A

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Office Action dated April 10, 2006, please amend the above-identified patent application as follows:

Amendments to the Specification begin on page 2 of this paper.

Amendments to the Claims are reflected in the listing of claims which begin on page 3 of this paper.

Remarks/Arguments begin on page 15 of this paper.

Amendments to the Specification:

Please replace the paragraph on page 1, starting at line 8, with the following amended paragraph:

--The invention described herein relates to and claims priority from the Provisional Patent Application ~~having U.S. Serial~~ Number 60/537,431 ~~(Attorney Docket No. SUN1P766P/P040534)~~, entitled "Method and Apparatus for Memory Management in a Multi-Processor Computer System", invented by Eric E. Lowe, filed on January 17, 2004. The aforementioned patent document is hereby incorporated by reference in its entirety for all purposes.--

Amendments to the Claims:

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A memory access method for use in a memory management unit (MMU) of a multi-processor computer system having a plurality of interconnected central processing units (CPU's) controlled by the same operating system and MMU, the method comprising:

A) initiating a memory access instruction concerning a selected virtual address having an associated context identifier;

B) searching a translation lookaside buffer (TLB) for a TLB entry having the selected virtual address;

C) wherein if a TLB entry having the selected virtual address is found in the TLB:

accessing an associated physical address translation from the TLB entry and executing the memory access instruction;

D) ~~where~~ wherein if the [[a]] TLB entry having the selected virtual address is NOT found in the TLB:

i) determining whether a translation table entry (TTE) corresponding with the selected virtual address is available in secondary memory assets of the system to have memory access instructions performed thereon, wherein said determining (D i) includes testing the associated context identifier to determine the availability of the TTE to have a memory access instruction performed thereon,

E) wherein if the TTE is available:

i) accessing the TTE from the secondary memory assets;

ii) updating at least one of the TLB and the secondary memory assets with information from the TTE; and

iii) returning to A) initiating a memory access instruction; and

F) wherein ~~said testing determines that~~ if the TTE is not available:

i) selectively pausing the method until the TTE becomes available; and

ii) returning to A) initiating a memory access instruction.

2. (Canceled)

3. (Currently Amended) The memory access method as in Claim [[2]] 1 wherein testing the associated context identifier is performed prior to accessing the TTE from the secondary memory assets.

4. (Original) The memory access method as in Claim 1 wherein updating at least one of the TLB and the secondary memory assets with information from the TTE includes updating with information including physical address information.

5. (Original) The memory access method as in Claim 1 wherein updating at least one of the TLB and the secondary memory assets with information from the TTE includes updating with information including attribute information.

6. (Original) The memory access method as in Claim 1 wherein when the selected virtual address is NOT found in the TLB, said D)(i) determining whether the TTE in secondary memory assets is available comprises:

invoking a TLB miss handler to obtain a TTE that corresponds to the selected virtual address having the associated context identifier, the miss handler conducting the operations of:

a) testing the context identifier, prior to accessing secondary memory assets, to determine if the TTE is available to have memory access instructions performed thereon,

wherein said testing determines that the TTE is available,

i) accessing the TTE from the secondary memory assets;

ii) updating the at least one of TLB and the secondary memory assets with information from the TTE as needed;

iii) returning to A) initiating a memory access instruction;

wherein said testing determines that the TTE is not available,

iv) invoking a miss exception handler to facilitate resolution of a miss exception event that resulted in said unavailability; and

v) returning to A) initiating a memory access instruction.

7. (Currently Amended) The method of Claim 6, wherein a)(iv) invoking a miss exception handler to resolve a miss exception event that resulted in said unavailability includes:

determining the nature of the miss exception event;

selectively pausing the operation of the miss exception handler ~~when necessary~~ until the miss exception event has been resolved;

facilitating the resolution of the miss exception event; and

returning to A) initiating a memory access instruction.

8. (Currently Amended) A computer readable media including computer program code for operating a memory management unit (MMU) of a multi-processor computer system having a plurality of interconnected central processing units (CPU's) controlled by the same operating system and MMU, the computer readable media including:

A) computer program code instructions for initiating a memory access instruction concerning a selected virtual address having an associated context identifier;

B) computer program code instructions for searching a translation lookaside buffer (TLB) for a TLB entry having the selected virtual address;

C) wherein the computer program code instructions find the TLB entry having the selected virtual address in the TLB, the code implements instructions for:

accessing an associated physical address translation from the TLB entry;

executing the memory access instruction using the associated physical address to complete a process;

D) wherein the computer program code instructions do NOT find the TLB entry having the selected virtual address in the TLB, the code implements instructions for invoking a TLB miss handler to obtain a <u>translation table entry (TTE)</u> [[TTE]] that corresponds to the selected virtual address and associated context identifier, the miss handler executing computer program code instructions for:

i) determining whether a TTE in secondary memory assets having the selected virtual address and associated context identifier is available to have memory access instructions performed thereon, <u>wherein said determining (D i) includes testing the associated context identifier to determine the availability of the TTE to have a memory access instruction performed thereon,</u>

E) wherein <u>if</u> the computer program code instructions determine that the TTE is available, the code implements:

i) computer program code instructions for accessing the TTE from the secondary memory assets;

ii) computer program code instructions for updating the at least one of the TLB and the secondary memory assets with information from the TTE; and

iii) computer program code instructions for returning to A) initiating a memory access instruction; and

F) wherein if the computer program code instructions determine that the TTE is not available, the code implements:

i) computer program code instructions for invoking a miss exception handler to resolve a miss exception event that resulted in said unavailability; and

ii) computer program code instructions for returning to A) initiating a memory access instruction.

9. (Original) The computer readable media including computer program code of Claim 8, wherein F)(i) computer program code instructions for invoking a miss exception handler to resolve a miss exception event includes:

computer program code instructions for determining the nature of the miss exception event;

computer program code instructions for selectively pausing the operation of the miss exception handler as necessary until the miss exception event has been resolved; and

computer program code instructions for resolving the miss exception event.

10. (Currently Amended) The computer readable media including computer program code of Claim 8, wherein F)(i) computer program code instructions for invoking a miss exception handler to resolve a miss exception event includes computer program code instructions for invoking a miss exception handler without issuing cross calls that effectively halt one or more ~~the operation of all~~ CPU's in the multi-processor computing system.

11. (Currently Amended) A computer system comprising:

a multi-processor computer system having a plurality of interconnected central processing units (CPU's) controlled by the same operating system;

each CPU having a memory cache configured to include a translation lookaside buffer (TLB) having entries configured to include virtual addresses and associated context identifiers for those virtual addresses;

secondary memory assets that include a translation storage buffer and page tables, said secondary memory assets resources including at least one translation table entry (TTE);

a memory management unit (MMU) that includes:

a TLB miss handler that enables:

searching the secondary memory assets to find a selected TTE that is related to a selected virtual address and associated context identifier when the selected virtual address and associated context identifier cannot be located in the TLB, and

testing the associated context identifier to determine if the TTE is available to have a memory access instruction executed thereon, wherein said testing occurs before the searching of the secondary memory assets; and

a miss exception handler that enables:

determining the nature of a TLB miss exception that renders the TTE unavailable;

determining that the miss exception has been resolved; and

selectively pausing the operation of the miss exception handler, if needed, until the miss exception is resolved.

12. (Currently Amended) The computer system of Claim 11 wherein the MMU is configured so that TLB misses and TLB miss exceptions are resolved without issuing cross calls that effectively halt one or more ~~the operation of all~~ CPU's in the multi-processor computing system.

13. (Currently Amended) A method of handling translation lookaside buffer (TLB) miss exceptions in a memory management unit of a multi-processor computer system having a plurality of Central Processing Units (CPU's) ~~CPU's~~, the method comprising:

initiating an access instruction;

determining that a TLB miss exception event has occurred, wherein determining that a TLB miss exception event has occurred includes testing a context identifier for the affected virtual address to determine whether a TLB miss exception has occurred;

invoking a miss exception handler;

determining the nature of the TLB miss exception event;

resolving the miss exception event; and

returning to ~~the operation of the memory management unit~~ initiating an access instruction.

14. (Original) The method of Claim 13 wherein the resolving of the miss exception event includes ~~selectively~~ pausing the miss exception handler ~~when necessary~~ until the miss exception event is resolved.

15. (Canceled)

16. (Original) The method of Claim 15 wherein determining the nature of the TLB miss exception event includes testing the context identifier for the affected virtual address to determine whether the TLB miss exception event results from one of an unassigned context identifier, a translation storage buffer (TSB) resizing operation that affects the virtual address, and an unmapping of a shared translation table entry (TTE) that is associated with the virtual address wherein the TTE comprises a shared memory resource; and

wherein resolving the miss exception events includes resolving each type of miss exception event in accordance with specified miss exception resolution protocol for each type of miss exception event.

17. (Original) The method of Claim 13 wherein determining the nature of the TLB miss exception event includes testing a context identifier for the affected virtual address to determine whether the TLB miss exception event results from one of an unassigned context identifier, a translation storage buffer (TSB) resizing operation that affects the virtual address, and an unmapping of a shared translation table entry (TTE) that is associated with the virtual address wherein the TTE comprises a shared memory resource; and

wherein resolving the miss exception events includes resolving each type of miss exception event in accordance with specified miss exception resolution protocol for each type of miss exception event.

18. (Currently Amended) The method of Claim 17 wherein said testing the context identifier determines that said unavailability results from a miss exception due to an unassigned context identifier, and

wherein resolving the unassigned context identifier miss exception event further includes the steps of:

assigning a context identifier value to a virtual address space that is to be associated with a process that contains the virtual address;

assigning selected portion of memory to a TSB that is to be associated with the virtual address space having the assigned context identifier;

updating at least one of secondary memory assets and the TLB with TSB and context identifier information; and

returning to ~~operation of the memory management unit~~ initiating an access instruction.

19. (Currently Amended) The method of Claim 17 wherein said testing the context identifier determines that said unavailability results from a miss exception due to a situation wherein the affected virtual address maps to a shared memory resource, and

wherein resolving the miss exception event further includes the steps of:

determining if the shared memory resource is locked;

if the shared memory resource is not locked,

returning to ~~operation of the memory management unit~~ initiating an access instruction; and

if the shared memory resource is locked,

pausing the exception handler until the shared memory resource becomes unlocked; and

returning to ~~operation of the memory management unit~~ initiating an access instruction when the shared memory resource becomes unlocked.

20. (Original) The method of Claim 19 wherein said shared memory resource comprises a translation table entry (TTE).

21. (Currently Amended) The method of Claim 17 wherein testing the context identifier determines that said unavailability results from a miss exception due to a situation wherein the virtual address is associated with a TSB that is undergoing a resizing operation; and

wherein resolving the miss exception event further includes the steps of:

A) determining if the virtual address space of the TSB undergoing resizing is locked;

1) in a case wherein the virtual address space of the TSB undergoing resizing is locked,

i) pausing the exception handler until the virtual address space of the TSB undergoing resizing becomes unlocked;

ii) locking the virtual address space of the TSB undergoing resizing;

2) in a case wherein the virtual address space of the TSB undergoing resizing is unlocked,

i) locking the virtual address space of the TSB undergoing resizing;

B) once the virtual address space of the TSB undergoing resizing has been locked by one of 1)(ii) and 2(i), determining whether the TSB undergoing resizing has been assigned a specified location in memory;

1) if the TSB undergoing resizing has been assigned a specified location in memory,

i) releasing the lock on the virtual address space of the TSB undergoing resizing; and

ii) returning to ~~operation of the memory management unit~~ initiating an access instruction;

2) if the TSB undergoing resizing has not been assigned a specified location in memory,

i) assigning a specified portion of memory to the TSB undergoing resizing;

ii) releasing the lock on the virtual address space of the TSB undergoing resizing; and

iii) returning to ~~operation of the memory management unit~~ initiating an access instruction.

22. (Currently Amended) A method of accessing translation table entries (TTE's) in secondary memory assets of a memory management unit of a multi-processor computer system, the method comprising:

A) requesting that a translation be found for a selected virtual address having an associated context identifier wherein the translation is to be found in a secondary memory asset of a memory management unit;

B) testing the associated context identifier, prior to searching a secondary memory asset, to determine whether a TTE corresponding to the virtual address and the associated context identifier is available to have a memory access instruction executed upon it;

1) wherein if testing determines the TTE is available to have a memory access instruction executed upon it:

locating the TTE using the virtual address and context identifier;

accessing the TTE from the secondary memory asset;

updating a translation lookaside buffer (TLB) and the secondary memory asset as needed;

returning to ~~operation of the memory management unit~~ initiating an access instruction;

2) wherein if testing determines the TTE is not available to have a memory access instruction executed upon it:

determining a source of unavailability;

resolving the unavailability; and

returning to ~~operation of the memory management unit~~ initiating an access instruction.

23. (Original) A method as in Claim 22 wherein the secondary memory assets include at least one translation storage buffer (TSB) and at least one page table,

wherein locating the TTE using the virtual address and context identifier include first searching the TSB for a TTE corresponding with the virtual address and associated context identifier and then if a TTE corresponding with the virtual address and associated context identifier is not found in the TSB, locating a TTE corresponding with the virtual address and associated context identifier in a page table;

and

wherein updating, when the TTE is located in a page table comprises updating the TLB to include the TTE and updating the TSB to include the TTE.

24. (Currently Amended) A method as in Claim 22 wherein the secondary memory assets include at least one translation storage buffer (TSB) and at least one page table, and

wherein resolving the unavailability includes:

determining the nature of the unavailability; and

where it is determined that the cause of the unavailability is a TSB resizing operation that affects the virtual address for which the translation is sought;

pausing until the TSB resizing operation is completed; and

returning to ~~operation of the memory management unit~~ initiating an access instruction.

25. (Currently Amended) A method as in Claim 22 wherein the secondary memory assets include at least one translation storage buffer (TSB) and at least one page table, and

wherein resolving the unavailability includes:

determining the nature of the unavailability; and

where it is determined that the cause of the unavailability is a demapping operation for a TTE that is shared by the virtual address for which the translation is sought,

pausing until the TTE demapping operation is completed; and

returning to ~~operation of the memory management unit~~ initiating an access instruction.

26. (Currently Amended) A method of accomplishing memory management of miss exceptions in a memory management unit of a multi-processor computer system, the method comprising;

determining that a miss exception event has occurred, wherein ~~the miss~~ the miss exception event concerns one of: an unassigned context identifier event, a memory access event changing a shared memory resource, and a TSB resizing event; and

resolving the miss exception event in accordance with a miss event resolution protocol suitable for resolving the received miss exception event.

27. (Original) The method of Claim 26 wherein said determining determines that the miss exception event comprises an instruction to change a translation table entry (TTE) that is shared by more than one virtual address space wherein each virtual address space has an associated context identifier; and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

identifying virtual address spaces that share the same TTE;

activating a lock for each virtual address space that shares the same TTE to prevent other processes from accessing the TTE while the lock is activated;

changing the corresponding context identifier for each virtual address space that shares the same TTE thereby making the associated TTE unavailable to have memory access instructions performed thereon;

performing the necessary changes on the TTE;

releasing the locks on each locked virtual address space; and

freeing the corresponding context identifiers for each affected virtual address space.

28. (Original) The method of Claim 26 wherein said determining determines that the miss exception event comprises a miss exception event comprises an instruction to resize a translation storage buffer (TSB); and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

activating a lock for the TSB to prevent other processes from accessing entries in the TSB while the lock is activated;

changing the corresponding context identifier to indicate that the TSB virtual address space is unavailable to have memory access instructions performed thereon;

resizing the TSB;

changing the corresponding context identifier back to its original configuration; and

releasing the lock on the TSB.

29. (Original) The method of Claim 26 wherein said determining determines that the miss exception event comprises an instruction concerning an unassigned context identifier; and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

assigning a context identifier for a virtual address space associated with a TSB; and

assigning a portion of memory to the TSB.

<u>REMARKS/ARGUMENTS</u>

It is respectfully submitted that the oath and declaration were submitted prior to an application number being assigned and therefore it is not necessary for Applicants to submit a new oath or declaration.

The Examiner's rejection of claims under 35 U.S.C. 112 and 103(a) is fully traversed below.

<u>Rejections of claims under 35 U.S.C. 112</u>

It is respectfully submitted that a Translation Lookaside Buffer (TLB) is searched for a TLB entry (see, for example, claim 1, operation B). Clearly, a TLB can be searched for a TLB entry. Further, claims 10 and 12 recite invoking a miss exception handler <u>without</u> issuing cross calls. As such, it is respectfully submitted that no additional explanation of how cross calls would be issued is needed.

Claims have been amended to recite "returning to a memory access instruction" (see, for example, operation 427 and 437 shown in Figures 4.2 and 4.3).

In addition, further minor corrections have been made to correct typographical and/or grammatical errors. Accordingly, it is respectfully requested that the Examiner withdraw the rejection under 35 U.S.C. § 112.

<u>Rejection of claims under 35 U.S.C. § 103(a)</u>

In the Office Action, the Examiner has noted that *Khalidi et al.* does not disclose a virtual address "having an associated context identifier" (Office Action, page 11). Nevertheless, the Examiner has asserted that *Khalidi et al.* teaches testing an associated context identifier to determine the availability of the TTE to have a memory instruction performed thereon. Clearly, the Examiner's rejection is improper because *khalidi et al.* cannot possibly teach testing a context identifier given that *Khalidi et al.* fails to teach a context identifier.

Moreover, it is respectfully submitted that neither *Khalidi et al.* nor Mohamed teach or suggest testing a context identifier associated with a virtual address. Claim 1 recites this feature and therefore is patentable over *Khalidi et al.* and *Mohamed.* Other independent claims recite this feature and therefore are patentable over the cited art.

CONCLUSION

Based on the foregoing, it is submitted that the claims are patentably distinct over the cited art of record. Additional limitations recited in the independent claims or the dependent claims are not further discussed because the limitations discussed above are sufficient to distinguish the claimed invention from the cited art. Accordingly, Applicant believes that all pending claims are allowable and respectfully requests a Notice of Allowance for this application from the Examiner.

Applicants hereby petition for an extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 500388 (Order No. SUN1P766). Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

/Rambusian/
Ramin Mahboubian
Reg. No. 44,890

June 19, 2006

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300